```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

K 16

```
**FILE**ID**NTODAPIO




NN      NN  TTTTTTTTTT    000000    DDDDDDDD      AAAAAA    PPPPPPPP      IIIIII      000000
NN      NN  TTTTTTTTTT    000000    DDDDDDDD      AAAAAA    PPPPPPPP      IIIIII      000000
NN      NN      TT      00      00  DD      DD  AA      AA  PP      PP      II      00      00
NN      NN      TT      00      00  DD      DD  AA      AA  PP      PP      II      00      00
NNNN    NN      TT      00    0000  DD      DD  AA      AA  PP      PP      II      00      00
NNNN    NN      TT      00    0000  DD      DD  AA      AA  PP      PP      II      00      00
NN  NN  NN      TT      00  00  00  DD      DD  AA      AA  PPPPPPPP        II      00      00
NN  NN  NN      TT      00  00  00  DD      DD  AA      AA  PPPPPPPP        II      00      00
NN    NNNN      TT      0000    00  DD      DD  AAAAAAAAAA  PP              II      00      00
NN    NNNN      TT      0000    00  DD      DD  AAAAAAAAAA  PP              II      00      00
NN      NN      TT      00      00  DD      DD  AA      AA  PP              II      00      00      ....
NN      NN      TT      00      00  DD      DD  AA      AA  PP              II      00      00      ....
NN      NN      TT        000000    DDDDDDDD    AA      AA  PP            IIIIII      000000        ....
NN      NN      TT        000000    DDDDDDDD    AA      AA  PP            IIIIII      000000        ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
Table of contents
```

```
    (2)        101         DECLARATIONS
    (3)        141         NT$TRANSMIT - SEND DAP MESSAGE
    (4)        322         NT$TRANSMIT_PKT - SEND DAP MESSAGE PACKET
    (6)        479         NT$INTERRUPT - SEND INTERRUPT MESSAGE
    (7)        554         NT$RECEIVE - RECEIVE DAP MESSAGE
    (8)        690         NT$STALLAST - RECEIVE AST ROUTINE
    (9)        775         PROCESS DAP MESSAGE PARSE FAILURE
    (10)       824         PROCESS DAP STATUS MESSAGE FROM FAL
    (11)       987         NT$RMT_xxx ERROR REPORTING ROUTINES
    (12)      1034         NT$LCL_xxx ERROR REPORTING ROUTINES
```

```
0000          $BEGIN  NTODAPIO,000,NK$NETWORK,<DAP MESSAGE I/O>
0000    
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                *
0000    ;*   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
0000    ;*   ALL RIGHTS RESERVED.                                                   *
0000    ;*                                                                          *
0000    ;*   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
0000    ;*   ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0000    ;*   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
0000    ;*   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
0000    ;*   OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
0000    ;*   TRANSFERRED.                                                           *
0000    ;*                                                                          *
0000    ;*   THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
0000    ;*   AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    ;*   CORPORATION.                                                           *
0000    ;*                                                                          *
0000    ;*   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
0000    ;*   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000    
0000    ;++
0000    ; Facility: RMS
0000    ;
0000    ; Abstract:
0000    ;
0000    ;       This module contains routines that transmit and receive DAP messages
0000    ;       to and from the File Access Listener (FAL) at the remote node.
0000    ;
0000    ; Environment: VAX/VMS, executive mode
0000    ;
0000    ; Author: James A. Krycka,      Creation Date:  09-DEC-1977
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V03-006 JAK0145         J A Krycka      12-APR-1984
0000    ;               Revise DAP message blocking algorithm in FAL$TRANSMIT to
0000    ;               eliminate an extra move of the data and rename the message
0000    ;               descriptors for clarity.
0000    ;
0000    ;       V03-005 JAK0139         J A Krycka      02-APR-1984
0000    ;               Support DAP buffer size up to 65535 bytes.
0000    ;
0000    ;       V03-004 JAK0133         J A Krycka      20-MAR-1984
0000    ;               Change a signed compare to an unsigned compare in NT$TRANSMIT.
0000    ;
0000    ;       V03-003 JEJ0002         J E Johnson     28-Feb-1984
0000    ;               Look at proper file block area (image or ppf) for possible
0000    ;               rundown in progress flag.
0000    ;
0000    ;       V03-002 JAK0119         J A Krycka      16-JUL-1983
```

```
                0000     58 ;             Add NT$LCL_JOP error routine.
                0000     59 ;
                0000     60 ;     Keep following revision history for technical documentation.
                0000     61 ;
                0000     62 ;     V02-034 TMK0002           Todd M. Katz     21-FEB-1982
                0000     63 ;             Whenever we are waiting in RM$STALL, we are waiting on the
                0000     64 ;             event flag associated with the transfer operation. This will
                0000     65 ;             be regardless of whether we are waiting for the transfer or the
                0000     66 ;             special receive operation to complete. If we are waiting for
                0000     67 ;             both operations to complete, and one of them does, then when
                0000     68 ;             we dismiss the AST we stall again waiting (on the transfer
                0000     69 ;             operation's event flag) for the second operation to complete.
                0000     70 ;             This will be regardless of whether the receive or the transfer
                0000     71 ;             operation completes first. In file transfer mode it will be
                0000     72 ;             possible that the receive associated with the first $PUT is
                0000     73 ;             associated with one event flag, while the transfer associated
                0000     74 ;             with an explicit $DISCONNECT is associated with another. This
                0000     75 ;             quite likely be the case when the RMS record operations are
                0000     76 ;             performed asynchronously. To prevent the process hang which
                0000     77 ;             will occur when the transfer operation completes before the
                0000     78 ;             receive and the receive AST is delivered while we are again
                0000     79 ;             waiting for the event flag associated with the transfer
                0000     80 ;             operation to be set, I have made two changes to the special
                0000     81 ;             receive AST synchronization code. First, when the special
                0000     82 ;             AST is posted, the event flag used will always the throw
                0000     83 ;             event flag (event flag# 31). Second, when the special receive
                0000     84 ;             AST is delivered, and it is determined within NT$STALLAST
                0000     85 ;             that a stall was explicitely requested, the event flag
                0000     86 ;             associated with the transfer operation will be specifically
                0000     87 ;             set preventing any possible hangs do to the receive and
                0000     88 ;             the transfer QIOs specifying different event flags.
                0000     89 ;
                0000     90 ;     V02-033 TMK0001           Todd M. Katz     19-FEB-1982
                0000     91 ;             In NT$STALLAST, if this AST is for a special receive we have
                0000     92 ;             not stalled for, and RMS rundown is in progress, then before
                0000     93 ;             dismissing the AST, set the IOREFN event flag. RMS rundown
                0000     94 ;             has done a $CANCEL on this channel with ASTs disabled, cleared
                0000     95 ;             this event flag, and is waiting for it to be set (indicating
                0000     96 ;             that all QIOs canceled have completed) before it can continue
                0000     97 ;             on and eventually close the file.
                0000     98 ;
                0000     99 ;--
```

```
               0000          .SBTTL  DECLARATIONS
               0000
               0000   ;
               0000   ; Include Files:
               0000   ;
               0000
               0000          $DAPPLGDEF                          ; Define DAP prologue symbols
               0000          $DAPHDRDEF                          ; Define DAP message header
               0000          $DAPCNFDEF                          ; Define DAP Configuration message
               0000          $DAPSTSDEF                          ; Define DAP Status message
               0000          $DAPFIDDEF                          ; Define DAP field ID symbols
               0000          $FABDEF                             ; Define File Access Block symbols
               0000          $IFBDEF                             ; Define IFAB symbols
               0000          $IMPDEF                             ; Define impure area definitions
               0000          $IODEF                              ; Define I/O function codes
               0000          $IRBDEF                             ; Define IRAB symbols
               0000          $NWADEF                             ; Define Network Work Area symbols
               0000          $PIODEF                             ; Define Process I/O Page symbols
               0000          $RMSDEF                             ; Define RMS completion codes
               0000          $RMSFALMSG                          ; Define FAL status codes
               0000
               0000   ;
               0000   ; Macros:
               0000   ;
               0000   ;      None
               0000   ;
               0000   ; Equated Symbols:
               0000   ;
               0000
     00000001  0000   RMS__FACILITY=1                            ; RMS facility code
               0000
               0000          ASSUME  DAP$Q_DCODE_FLG EQ 0
               0000          ASSUME  NWA$Q_FLG EQ 0
               0000
               0000   ;
               0000   ; Own Storage:
               0000   ;
               0000   ;      None
               0000   ;
```

```
0000            .SBTTL  NT$TRANSMIT - SEND DAP MESSAGE
0000
0000    ;++
0000    ; NT$TRANSMIT - sends the specified DAP message to the remote FAL.
0000    ;       DAP outbound message blocking is supported. Consequently, the new
0000    ;       message wil be buffered (blocked with others) and delivered later
0000    ;       unless the NWA$V_LAST_MSG bit is set or DAP messsage blocking is
0000    ;       disabled.
0000    ;
0000    ; Calling Sequence:
0000    ;
0000    ;       BSBW    NT$TRANSMIT
0000    ;
0000    ; Input Parameters:
0000    ;
0000    ;       R7      NWA (=DAP) address
0000    ;       R8      FAB/RAB address
0000    ;       R9      IFAB/IRAB address
0000    ;       R10     FWA/IFAB address
0000    ;       R11     Impure Area address
0000    ;
0000    ; Implicit Inputs:
0000    ;
0000    ;       DAP$V_LENGTH
0000    ;       DAP$V_MSGBLK
0000    ;       NWA$Q_BLD
0000    ;       NWA$Q_XMT
0000    ;       NWA$W_DAPBUFSIZ
0000    ;       NWA$V_FLUSH_BUF
0000    ;       NWA$V_LAST_MSG
0000    ;       NWA$V_RCVAST
0000    ;
0000    ; Output Parameters:
0000    ;
0000    ;       R0      Status code (RMS)
0000    ;       R1-R3   Destroyed
0000    ;       AP      Destroyed
0000    ;
0000    ; Implicit Outputs:
0000    ;
0000    ;       DAP$L_DCODE_STS (on error)
0000    ;       IFB$L_IOS/IRB$L_IOS
0000    ;       NWA$Q_XMT
0000    ;       NWA$V_FLUSH_BUF cleared
0000    ;       NWA$V_LAST_MSG cleared
0000    ;
0000    ; Completion Codes:
0000    ;
0000    ;       Standard RMS completion codes
0000    ;
0000    ; Side Effects:
0000    ;
0000    ;       None
0000    ;
0000    ;--
0000
0000    NT$TRANSMIT::                           ; Entry point
```

```
             30   BB  0000             PUSHR   #^M<R4,R5>                  ; Save registers
                      0002
                      0002 ;
                      0002 ; Check for possible Status message returned by FAL in response to a previous
                      0002 ; DAP message sent.
                      0002 ;
                      0002
     09 67   04   E1  0002             BBC     #NWA$V_RCVAST,(R7),5$       ; Branch if no DAP message received
        0174      30  0006             BSBW    NT$RECEIVE                  ; Process the (Status) message
       03 50      E8  0009             BLBS    R0,5$                       ; Continue only if it was a warning or
        008F      31  000C             BRW     EXIT                        ;  success Status message (which should
                      000F                                                 ;  not occur), else exit
                      000F
                      000F ;
                      000F ; Process flush-blocked-messages-only option.
                      000F ;
                      000F
        67   01   E5  000F 5$:         BBCC    #NWA$V_FLUSH_BUF,(R7),-     ; Branch if flush option is disabled
             09       0012                     10$                         ;  (and initialize for next time thru)
      00E8 C7     D5  0013             TSTL    NWA$Q_XMT(R7)               ; Branch if there are blocked DAP
           6C     12  0017             BNEQ    40$                         ;  messages in buffer (explicit flush)
        007F      31  0019             BRW     SUC                         ; Else, there is nothing to do
                      001C
                      001C ;
                      001C ; Does partner support message blocking?
                      001C ;
                      001C
                      001C 10$:        $TSTPT  NTDAP_ENC                   ;
             12   E0  0022             BBS     #DAP$V_MSGBLK,-             ; Branch if message blocking is
     03 28 A7         0024                     DAP$Q_SYSCAP(R7),15$        ;  enabled
        0068      31  0027             BRW     50$                         ; Else send message immediately
                      002A
                      002A ;
                      002A ; Are there any messages in XMT buffer waiting to go?
                      002A ;
                      002A
      00E8 C7     D5  002A 15$:        TSTL    NWA$Q_XMT(R7)               ; Branch if there are no blocked DAP
           20     13  002E             BEQL    20$                         ;  messages in buffer
                      0030
                      0030 ;
                      0030 ; Will new message fit in XMT buffer with other messages?
                      0030 ;
                      0030
      00F0 C7     C1  0030             ADDL3   NWA$Q_BLD(R7),-             ; Compute total # bytes if new message
 50   00E8 C7         0034                     NWA$Q_XMT(R7),R0            ;  were added to transmit buffer
 51   00CA C7     3C  0038             MOVZWL  NWA$W_DAPBUFSIZ(R7),R1      ; Prepare for longword compare as size
                      003D                                                 ;  in R0 may slightly exceed 64K
     51    50     D1  003D             CMPL    R0,R1                       ; Branch if new message can be blocked
           34     1B  0040             BLEQU   30$                         ;  with the others
                      0042
                      0042 ;
                      0042 ; Flush XMT buffer to make room for new message.
                      0042 ;
                      0042
 54   00E8 C7     7D  0042             MOVQ    NWA$Q_XMT(R7),R4            ; Put buffer descriptor in <R4,R5>
           5C     10  0047             BSBB    NT$TRANSMIT_PKT             ; Send blocked DAP messages
        52 50     E9  0049             BLBC    R0,EXIT                     ; Branch on failure
      00E8 C7    D4  004C   255             CLRL    NWA$Q_XMT(R7)              ; Initialize byte count in descriptor
                     0050   256
                     0050   257 ;
                     0050   258 ; Determine whether new message should be sent now or deferred (bufferred).
                     0050   259 ;
                     0050   260
            09   E1  0050   261 20$:        BBC     #<DAP$V_LENGTH+8>,-        ; Branch if this message does not allow
   3C 00F4 D7        0052   262                     @NWA$Q_BLD+4(R7),50$       ;  additional blocking (implicit flush)
    38 67    00  E0  0056   263             BBS     #NWA$V_LAST_MSG,(R7),50$; Branch if this is last message of
                     005A   264                                                ;  sequence (explicit flush)
                     005A   265
                     005A   266 ;
                     005A   267 ; Move new message to XMT buffer, or simply map message if BLD and XMT buffers
                     005A   268 ; are aligned.
                     005A   269 ;
                     005A   270
      00F0 C7    D0  005A   271             MOVL    NWA$Q_BLD(R7),-            ; Update byte count in descriptor
      00E8 C7        005E   272                     NWA$Q_XMT(R7)              ;
      00F4 C7    D1  0061   273             CMPL    NWA$Q_BLD+4(R7),-          ; Eliminate move of data if BLD and XMT
      00EC C7        0065   274                     NWA$Q_XMT+4(R7)            ;  buffer addresses coincide
           31    13  0068   275             BEQL    SUC                        ;
      00F0 C7    28  006A   276             MOVC3   NWA$Q_BLD(R7),-            ; Copy the message
      00F4 D7        006E   277                     @NWA$Q_BLD+4(R7),-         ;
      00EC D7        0071   278                     @NWA$Q_XMT+4(R7)           ;
           25    11  0074   279             BRB     SUC                        ;
                     0076   280
                     0076   281 ;
                     0076   282 ; Append new message to XMT buffer which already abuts XMT buffer.
                     0076   283 ;
                     0076   284
    00E8 C7    50  D0  0076   285 30$:        MOVL    R0,NWA$Q_XMT(R7)           ; Update byte count in descriptor
                     007B   286
                     007B   287 ;
                     007B   288 ; Determine whether new message should be sent now or deferred (bufferred).
                     007B   289 ;
                     007B   290
            09   E1  007B   291             BBC     #<DAP$V_LENGTH+8>,-        ; Branch if this message does not allow
   04 00F4 D7        007D   292                     @NWA$Q_BLD+4(R7),40$       ;  additional blocking (implicit flush)
    16 67    00  E1  0081   293             BBC     #NWA$V_LAST_MSG,(R7),SUC; Fall thru if this is last message of
                     0085   294                                                ;  sequence (explicit flush)
                     0085   295
                     0085   296 ;
                     0085   297 ; Send blocked messages from XMT buffer.
                     0085   298 ;
                     0085   299
 54   00E8 C7    7D  0085   300 40$:        MOVQ    NWA$Q_XMT(R7),R4           ; Put buffer descriptor in <R4,R5>
           19    10  008A   301             BSBB    NT$TRANSMIT_PKT            ; Send blocked DAP messages
      00E8 C7    D4  008C   302             CLRL    NWA$Q_XMT(R7)              ; Initialize byte count descriptor
           0C    11  0090   303             BRB     EXIT                       ;
                     0092   304
                     0092   305 ;
                     0092   306 ; Send new message from BLD buffer.
                     0092   307 ;
                     0092   308
 54   00F0 C7    7D  0092   309 50$:        MOVQ    NWA$Q_BLD(R7),R4           ; Put buffer descriptor in <R4,R5>
           0C    10  0097   310             BSBB    NT$TRANSMIT_PKT            ; Send the message
           03    11  0099   311             BRB     EXIT                       ;
```

```
                   009B   312
                   009B   313 ;
                   009B   314 ; Return to caller.
                   009B   315 ;
                   009B   316
                   009B   317 SUC:    RMSSUC                          ; Return success
                   009E   318 EXIT:   $CLRBIT #NWA$V_LAST_MSG,(R7)    ; Clear last-message-to-block flag
          30   BA  00A2   319         POPR    #^M<R4,R5>              ; Restore registers
               05  00A4   320         RSB                             ; Exit with RMS code in R0
```

```
00A5          .SBTTL  NT$TRANSMIT_PKT - SEND DAP MESSAGE PACKET
00A5
00A5    ;++
00A5    ; NT$TRANSMIT_PKT - sends the specified packet of DAP messages to the remote
00A5    ;       FAL.
00A5    ;
00A5    ; Calling Sequence:
00A5    ;
00A5    ;       BSBW    NT$TRANSMIT_PKT
00A5    ;
00A5    ; Input Parameters:
00A5    ;
00A5    ;       R4      Buffer size
00A5    ;       R5      Buffer address
00A5    ;       R7      NWA (=DAP) address
00A5    ;       R8      FAB/RAB address
00A5    ;       R9      IFAB/IRAB address
00A5    ;       R10     FWA/IFAB address
00A5    ;       R11     Impure Area address
00A5    ;
00A5    ; Implicit Inputs:
00A5    ;
00A5    ;       NWA$W_DAPBUFSIZ
00A5    ;       NWA$Q_RCV
00A5    ;       NWA$V_RCVQIO
00A5    ;
00A5    ; Output Parameters:
00A5    ;
00A5    ;       R0      Status code (RMS)
00A5    ;       R1-R3   Destroyed
00A5    ;       AP      Destroyed
00A5    ;
00A5    ; Implicit Outputs:
00A5    ;
00A5    ;       DAP$L_DCODE_STS (on error)
00A5    ;       DAP$Q_MSG_BUF1
00A5    ;       IFB$L_IOS/IRB$L_IOS
00A5    ;       NWA$V_RCVQIO
00A5    ;
00A5    ; Completion Codes:
00A5    ;
00A5    ;       Standard RMS completion codes
00A5    ;
00A5    ; Side Effects:
00A5    ;
00A5    ;       None
00A5    ;
00A5    ;--
00A5
00A5          ASSUME  IFB$L_IOS  EQ IRB$L_IOS
00A5
00A5    NT$TRANSMIT_PKT::                       ; Entry point
00A5          $TSTPT  NTDAP_XMT                 ;
00AB
00AB    ;+
00AB    ; Issue special receive QIO with AST if none has been posted, to guarantee that
00AB    ; there is always a receive posted to accept a possible error response (DAP
                         00AB   379 ; Status message) from FAL. If all goes well, then this receive will be used
                         00AB   380 ; to accept the normal DAP message response from FAL after the transmit message
                         00AB   381 ; sequence is complete.
                         00AB   382 ;
                         00AB   383 ; Note: This is termed a special receive QIO because (1) the NWA is used to
                         00AB   384 ;       store the I/O status block instead of the IFAB, and (2) NT$STALLAST is
                         00AB   385 ;       used to process the AST instead of RM$STALLAST. These changes were
                         00AB   386 ;       necessary to allow both a transmit and a receive QIO request to be
                         00AB   387 ;       outstanding on the same channel in RMS.
                         00AB   388 ;
                         00AB   389 ; Note: Always use the IFAB address as the AST parameter because the the IFAB is
                         00AB   390 ;       guaranteed to be around for both IFAB and IRAB based operations, as this
                         00AB   391 ;       receive may be posted in an IRAB context ($PUT) and completed in an IFAB
                         00AB   392 ;       context ($CLOSE).
                         00AB   393 ;-
                         00AB   394
     33 67     03   E0   00AB   395         BBS     #NWA$V_RCVQIO,(R7),10$   ; Branch if special receive is posted
        08 A7       D5   00AF   396         TSTL    DAP$Q_MSG_BUF1(R7)       ; Branch if receive buffer is in use
           2E       12   00B2   397         BNEQ    10$                      ;  indicating that a receive message
                         00B4   398                                          ;  sequence is in progress
           6C       10   00B4   399         BSBB    SETUP_QIO_PARAM          ; Return with IFAB address, EFN #, and
                         00B6   400                                          ;  channel # in R1-R3, respectively
                         00B6   401         $QIO_S-                          ; Issue receive QIO request
                         00B6   402                 EFN=#IMP$C_ASYQIOEFN-    ;  Use throw away event flag
                         00B6   403                 CHAN=R3-                 ;  Channel #
                         00B6   404                 FUNC=#IO$_READVBLK-      ;  Function code
                         00B6   405                 IOSB=NWA$Q_IOSB(R7)-     ;  Receive I/O status block address
                         00B6   406                 ASTADR=W^NT$STALLAST-    ;  AST routine address
                         00B6   407                 ASTPRM=R1-               ;  Always use IFAB address
                         00B6   408                 P1=@NWA$Q_RCV+4(R7)-     ;  Buffer address
                         00B6   409                 P2=NWA$W_DAPBUFSIZ(R7)   ;  Buffer size
        38 50       E9   00DB   410         BLBC    R0,ERRSYS                ; Branch on failure
                         00DE   411         $SETBIT #NWA$V_RCVQIO,(R7)       ; Set special receive QIO posted flag
                         00E2   412
                         00E2   413 ;+
                         00E2   414 ; Issue transmit QIO with AST.
                         00E2   415 ; If outbound DAP message blocking is in effect, then more than one DAP message
                         00E2   416 ; may be transmitted via this QIO.
                         00E2   417 ;-
                         00E2   418
           3E       10   00E2   419 10$:    BSBB    SETUP_QIO_PARAM          ; Return with IFAB address, EFN #, and
                         00E4   420                                          ;  channel # in R1-R3, respectively
                         00E4   421         $QIO_S-                          ; Issue transmit QIO request
                         00E4   422                 EFN=R2-                  ;  Event flag #
                         00E4   423                 CHAN=R3-                 ;  Channel #
                         00E4   424                 FUNC=#IO$_WRITEVBLK-     ;  Function code
                         00E4   425                 IOSB=IFB$L_IOS(R9)-      ;  IFAB/IRAB IOSB address
                         00E4   426                 ASTADR=L^RM$STALLAST-    ;  AST routine address
                         00E4   427                 ASTPRM=R9-               ;  IFAB/IRAB address
                         00E4   428                 P1=(R5)-                 ;  Buffer address
                         00E4   429                 P2=R4                    ;  Buffer size
        0D 50       E9   0106   430         BLBC    R0,ERRSYS                ; Branch on failure
     00000000'EF    16   0109   431         JSB     RM$STALL                 ; Await completion
        04 50       E9   010F   432         BLBC    R0,ERRSYS                ; Branch on failure
                         0112   433         RMSSUC                           ; Return success
              05         0115   434         RSB                              ; Return with RMS code in R0
                         0116   435
```

```
                        0116   ;+
                        0116   ; Common code for transmit, interrupt, and receive routines to process a QIO
                        0116   ; failure.
                        0116   ;-
                        0116
                        0116   ERRSYS: RMSERR  SYS,R1                          ; Set default RMS code
         FEE2'    30    011B           BSBW    RM$MAPERR                       ; Map SS code into RMS code and
                        011E                                                   ;  put SS code in STV field
         18 A7    D4    011E           CLRL    DAP$L_DCODE_STS(R7)             ; Zero message parse status code to
                        0121                                                   ;  indicate that this error was not
                        0121                                                   ;  generated from a Status message
                  05    0121           RSB                                     ; Return with RMS code in R0
```

```
                          0122   449 ;++
                          0122   450 ; This is a QIO support routine that obtains the IFAB address, event flag
                          0122   451 ; number, and channel number to use. This routine uses the fact that an IRAB
                          0122   452 ; block ID is even and an IFAB block ID is odd.
                          0122   453 ;
                          0122   454 ; Input Parameters:
                          0122   455 ;
                          0122   456 ;          R9       IFAB/IRAB address
                          0122   457 ;
                          0122   458 ; Output Parameters:
                          0122   459 ;
                          0122   460 ;          R1       IFAB address
                          0122   461 ;          R2       Event flag #
                          0122   462 ;          R3       Channel #
                          0122   463 ;--
                          0122   464
                          0122   465 SETUP_QIO_PARAM:                                 ; Control point
  00000000'EF    16       0122   466          JSB      RM$SETEFN                      ; Request event flag # to use
       52   8E   D0       0128   467          MOVL     (SP)+,R2                       ;  and store it
                          012B   468
                          012B   469          ASSUME   IFB$B_BID EQ IRB$B_BID
                          012B   470          ASSUME   <IFB$C_BID&1> EQ 1
                          012B   471          ASSUME   <IRB$C_BID&1> EQ 0
                          012B   472
       51   59   D0       012B   473          MOVL     R9,R1                          ; Get IFAB/IRAB address
    03 08   A9   E8       012E   474          BLBS     IFB$B_BID(R9),10$              ; Branch if this is an IFAB
       51   69   D0       0132   475          MOVL     IRB$L_IFAB_LNK(R9),R1          ; Get IFAB address from IRAB
  53    20  A1   3C       0135   476 10$:     MOVZWL   IFB$W_CHNL(R1),R3              ; Get channel #
                 05       0139   477          RSB                                     ; Exit
```

```
0013A         .SBTTL  NT$INTERRUPT - SEND INTERRUPT MESSAGE
013A
013A    ;++
013A    ; NT$INTERRUPT - sends the specified DAP message to the remote FAL as an
013A    ;       interrupt message.
013A    ;
013A    ; Calling Sequence:
013A    ;
013A    ;       BSBW    NT$INTERRUPT
013A    ;
013A    ; Input Parameters:
013A    ;
013A    ;       R7      NWA (=DAP) address
013A    ;       R8      FAB/RAB address
013A    ;       R9      IFAB/IRAB address
013A    ;       R10     IFAB address
013A    ;       R11     Impure Area address
013A    ;
013A    ; Implicit Inputs:
013A    ;
013A    ;       NWA$Q_XMT
013A    ;
013A    ; Output Parameters:
013A    ;
013A    ;       R0      Status code (RMS)
013A    ;       R1-R3   Destroyed
013A    ;       AP      Destroyed
013A    ;
013A    ; Implicit Outputs:
013A    ;
013A    ;       DAP$L_DCODE_STS (on error)
013A    ;       IFB$L_IOS/IRB$L_IOS
013A    ;
013A    ; Completion Codes:
013A    ;
013A    ;       Standard RMS completion codes
013A    ;
013A    ; Side Effects:
013A    ;
013A    ;       None
013A    ;
013A    ;--
013A
013A            ASSUME  IFB$L_IOS  EQ IRB$L_IOS
013A
013A    NT$INTERRUPT::                                  ; Entry point
013A            $TSTPT  NTDAP_XMT                       ;
0140
0140    ;+
0140    ; Issue interrupt QIO with AST.
0140    ;
0140    ; Note: It is not necessary to issue a special receive QIO (like NT$TRANSMIT_PKT
0140    ;       does) because FAL is not in a state where it is trying to send RMS a
0140    ;       message. Rather, FAL is waiting for this interrupt message to tell it
0140    ;       what to do next.
0140    ;-
0140
```

```
          E0  10  0140   536          BSBB    SETUP_QIO_PARAM             ; Return with IFAB address, EFN #, and
                  0142   537                                              ;   channel # in R1-R3, respectively
                  0142   538          $QIO_S-                             ; Issue interrupt QIO request
                  0142   539                  EFN=R2-                     ;   Event flag #
                  0142   540                  CHAN=R3-                    ;   Channel #
                  0142   541                  FUNC=#IO$_WRITEVBLK!IO$M_INTERRUPT- ; Function code
                  0142   542                  IOSB=IFB$L_IOS(R9)-         ;   IFAB/IRAB IOSB address
                  0142   543                  ASTADR=L^RM$STALLAST-       ;   AST routine address
                  0142   544                  ASTPRM=R9-                  ;   IFAB/IRAB address
                  0142   545                  P1=@NWA$Q_XMT+4(R7)-        ;   Buffer address
                  0142   546                  P2=NWA$Q_XMT(R7)            ;   Buffer size
       0D 50  E9  016A   547          BLBC    R0,ERRSYS1                  ; Branch on failure
   00000000'EF  16  016D   548          JSB     RM$STALL                    ; Await completion
       04 50  E9  0173   549          BLBC    R0,ERRSYS1                  ; Branch on failure
                  0176   550          RMSSUC                              ; Return success
              05  0179   551          RSB                                 ; Return with RMS code in R0
        FF99  31  017A   552 ERRSYS1:BRW     ERRSYS                      ; Branch aid
```

```
017D            .SBTTL  NT$RECEIVE - RECEIVE DAP MESSAGE
017D
017D    ;++
017D    ; NT$RECEIVE - accepts the next DAP message packet from the remote FAL.
017D    ;       DAP inbound message blocking is supported. Consequently, the next
017D    ;       message processed may already be in the receive buffer (blocked with
017D    ;       others) unless DAP message blocking is disabled.
017D    ;
017D    ; Calling Sequence:
017D    ;
017D    ;       BSBW    NT$RECEIVE
017D    ;
017D    ; Input Parameters:
017D    ;
017D    ;       R7      NWA (=DAP) address
017D    ;       R8      FAB/RAB address
017D    ;       R9      IFAB/IRAB address
017D    ;       R10     FWA/IFAB address
017D    ;       R11     Impure Area address
017D    ;
017D    ; Implicit Inputs:
017D    ;
017D    ;       DAP fields
017D    ;       DAP$V_VAXVMS
017D    ;       NWA$Q_IOSB
017D    ;       NWA$Q_RCV
017D    ;       NWA$V_NODECODE
017D    ;       NWA$V_RCVAST
017D    ;       NWA$V_RCVQIO
017D    ;
017D    ; Output Parameters:
017D    ;
017D    ;       R0      Status code (RMS)
017D    ;       R1-R3   Destroyed
017D    ;       AP      Destroyed
017D    ;
017D    ; Implicit Outputs:
017D    ;
017D    ;       DAP$L_DCODE_STS (on error)
017D    ;       DAP fields
017D    ;       IFB$L_IOS/IRB$L_IOS
017D    ;       NWA$L_THREAD
017D    ;       NWA$W_DAPBUFSIZ
017D    ;       NWA$V_NODECODE cleared
017D    ;       NWA$V_RCVAST
017D    ;       NWA$V_RCVQIO
017D    ;       NWA$V_RCVSTALL
017D    ;
017D    ; Completion Codes:
017D    ;
017D    ;       Standard RMS completion codes
017D    ;
017D    ; Side Effects:
017D    ;
017D    ;       None
017D    ;
017D    ;--
```

```
                        017D  611
                        017D  612          ASSUME  IFB$L_IOS  EQ IRB$L_IOS
                        017D  613
                        017D  614 NT$RECEIVE::                              ; Entry point
                        017D  615          $TSTPT  NTDAP_RCV                ;
         08 A7     D5   0183  616          TSTL    DAP$Q_MSG_BUF1(R7)       ; Branch if there is a blocked DAP
            71     12   0186  617          BNEQ    DECODE_DAP_MSG           ;  message in receive buffer to process
                        0188  618
                        0188  619 ;+
                        0188  620 ; Check for special receive QIO posted (by NT$TRANSMIT_PKT).
                        0188  621 ; If DAP inbound message blocking is in effect, then more than one DAP message
                        0188  622 ; may be received via this QIO.
                        0188  623 ;-
                        0188  624
   2C 67    03     E1   0188  625          BBC     #NWA$V_RCVQIO,(R7),20$   ; Branch if special receive not posted
   0F 67    04     E0   018C  626          BBS     #NWA$V_RCVAST,(R7),10$   ; Branch if special received completed
                        0190  627          $SETBIT #NWA$V_RCVSTALL,(R7)     ; Set flag to resume thread after stall
 00FC C7    59     D0   0194  628          MOVL    R9,NWA$L_THREAD(R7)      ; Save IFAB/IRAB address that we are
                        0199  629                                           ;  stalling on for use by NT$STALLAST
                        0199  630                                           ;  before it branches to RM$THREADGO
  00000000'EF      16   0199  631          JSB     RM$STALL                 ; Await completion of special receive
                        019F  632                                           ; Note: R0 contains garbage on return
                        019F  633 10$:     $CLRBIT #NWA$V_RCVQIO,(R7)       ; Clear receive posted flag
                        01A3  634          $CLRBIT #NWA$V_RCVAST,(R7)       ; Clear receive AST delivered flag
50    00D8 C7      3C   01A7  635          MOVZWL  NWA$Q_IOSB(R7),R0        ; Get status code
         73 50     E9   01AC  636          BLBC    R0,ERRSYS2               ; Branch on failure
       00DA C7     3C   01AF  637          MOVZWL  NWA$Q_IOSB+2(R7),-       ; Store # bytes received in
       00E0 C7          01B3  638                  NWA$Q_RCV(R7)            ;  descriptor
            3B     11   01B6  639          BRB     30$                      ; Join common code
                        01B8  640
                        01B8  641 ;+
                        01B8  642 ; Issue receive QIO with AST unless there is a blocked message to process.
                        01B8  643 ; If DAP inbound message blocking is in effect, then more than one DAP message
                        01B8  644 ; may be received via this QIO.
                        01B8  645 ;-
                        01B8  646
          FF67     30   01B8  647 20$:     BSBW    SETUP_QIO_PARAM          ; Return with IFAB address, EFN #, and
                        01BB  648                                           ;  channel # in R1-R3, respectively
                        01BB  649          $QIO_S-                          ; Receive DAP messages(s) from FAL
                        01BB  650                  EFN=R2-                  ;  Event flag #
                        01BB  651                  CHAN=R3-                 ;  Channel #
                        01BB  652                  FUNC=#IO$_READVBLK-      ;  Function code
                        01BB  653                  IOSB=IFB$L_IOS(R9)-      ;  IFAB/IRAB IOSB address
                        01BB  654                  ASTADR=L^RM$STALLAST-    ;  AST routine address
                        01BB  655                  ASTPRM=R9-               ;  NWA address plus flag
                        01BB  656                  P1=@NWA$Q_RCV+4(R7)-     ;  Buffer address
                        01BB  657                  P2=NWA$W_DAPBUFSIZ(R7)   ;  Buffer size
         3E 50     E9   01E1  658          BLBC    R0,ERRSYS2               ; Branch on failure
  00000000'EF      16   01E4  659          JSB     RM$STALL                 ; Await completion
         35 50     E9   01EA  660          BLBC    R0,ERRSYS2               ; Branch on failure
         0E A9     3C   01ED  661          MOVZWL  IFB$L_IOS+2(R9),-        ; Store # bytes received in
       00E0 C7          01F0  662                  NWA$Q_RCV(R7)            ;  descriptor
       00E0 C7     7D   01F3  663 30$:     MOVQ    NWA$Q_RCV(R7),-          ; Copy descriptor to DAP control
         08 A7          01F7  664                  DAP$Q_MSG_BUF1(R7)       ;  block
                        01F9  665
                        01F9  666 ;+
                        01F9  667 ; Decode next DAP message received and process any message parsing failure or
```

```
NTO
VAX

Mac
---
_$2
_$2
_$2
TOT

208

The

MAC
```

```
                        01F9    668 ; receipt of a DAP Status message.
                        01F9    669 ;-
                        01F9    670
                        01F9    671 DECODE_DAP_MSG:                                  ; Parse next DAP message
                        01F9    672             $TSTPT  NTDAP_DEC                    ;
      1B 67    02   E4  01FF    673             BBSC    #NWA$V_NODECODE,(R7),10$; Branch if message is not to be parsed
                        0203    674                                                  ;  (and initialize for next time thru)
                        0203    675             $SETBIT #DAP$K_STS_MSG,-             ; Flag Status message as valid
                        0203    676                     DAP$L_MSG_MASK(R7)           ;  to receive from FAL
               57   DD  0208    677             PUSHL   R7                           ; Push address of DAP control block
   0000'CF     01   FB  020A    678             CALLS   #1,W^NT$DECODE_MSG           ; Decode message into DAP control block
         1C A7      D4  020F    679             CLRL    DAP$L_MSG_MASK(R7)           ; Clear valid message flags
                        0212    680                                                  ;  (initialize for next time thru)
         5E 50      E9  0212    681             BLBC    R0,PARSE_FAILURE             ; Branch if message parse failed
         1A A7      91  0215    682             CMPB    DAP$B_DCODE_MSG(R7),-        ; Branch if DAP Status message was
               09       0218    683                     #DAP$K_STS_MSG               ;  received
               03   12  0219    684             BNEQ    10$
             0095   31  021B    685             BRW     STATUS_RETURN                ;
                        021E    686 10$:        RMSSUC                               ; Return success
               05       0221    687             RSB                                  ; Return with RMS code in R0
             FEF1   31  0222    688 ERRSYS2:BRW     ERRSYS                           ; Branch aid
```

```
0225          .SBTTL  NT$STALLAST - RECEIVE AST ROUTINE
0225
0225   ;++
0225   ; NT$STALLAST - is the receive AST routine for a (special) receive posted
0225   ;       early by NT$TRANSMIT_PKT.
0225   ;
0225   ;       Note carefully the following:
0225   ;       The call to RM$CHKAST must be a BSBW and immediatedly follow the entry
0225   ;       mask. Furthermore, if ASTS are inhibited (as is the case after RMS is
0225   ;       entered but before the first call to RM$STALL), then this AST will be
0225   ;       requeued instead of returning control to the instruction following the
0225   ;       'BSBW RM$CHKAST' instruction. Then after RM$STALL is called, the AST
0225   ;       will be delivered and control will return after the 'BSBW RM$CHKAST'
0225   ;       instruction.
0225   ;
0225   ; Calling Sequence:
0225   ;
0225   ;       CALLS   #5,NT$STALLAST  (invoked by VAX/VMS as an AST routine)
0225   ;
0225   ; Input Parameters:
0225   ;
0225   ;       4(AP)   IFAB address
0225   ;
0225   ; Implicit Inputs:
0225   ;
0225   ;       Contents of IFAB
0225   ;       IFB$L_NWA_PTR
0225   ;       NWA$L_THREAD
0225   ;       NWA$V_RCVSTALL
0225   ;       impure area
0225   ;
0225   ; Output Parameters:
0225   ;
0225   ;       R0      Set to contents of 1st word of I/O status block
0225   ;       R1-R3   Destroyed
0225   ;       R4-R11  Contents before stall
0225   ;       AP      Destroyed
0225   ;       SP      Address of stack having same contents as before stall
0225   ;       PC      Restored to return in line after call to RM$STALL
0225   ;
0225   ; Implicit Outputs:
0225   ;
0225   ;       IMP$L_SAVED_SP set appropriately for new stack
0225   ;       IMP$V_AST set
0225   ;       NWA$V_RCVAST set
0225   ;       NWA$V_RCVSTALL cleared
0225   ;
0225   ; Completion Codes:
0225   ;
0225   ;       System service status code from first word of I/O status block
0225   ;
0225   ; Side Effects:
0225   ;
0225   ;       The AST may be requeued (by RM$CHKAST).
0225   ;       RMS will be running at AST level on exit.
0225   ;       Secondary user structures require reprobing before use.
0225   ;       Absolute stack addresses will be different on exit.
```

```
                              0225  ;       If RMS rundown is in progress, the I/O rundown event flag will be set.
                              0225  ;
                              0225  ;--
                              0225
                        0FFC  0225          .ENTRY   NT$STALLAST,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                   FDD6'  30  0227          BSBW     RM$CHKAST               ; Check for ASTS inhibited
            57   3C A9    D0  022A          MOVL     IFB$L_NWA_PTR(R9),R7    ; Get address of NWA
                              022E                                       ; Note: R9 always contains IFAB address
                              022E          $SETBIT  #NWA$V_RCVAST,(R7)      ; Set receive AST delivered flag
         20 67    05      E4  0232          BBSC     #NWA$V_RCVSTALL,(R7),30$; Branch if waiting in RM$STALL
                              0236                                       ;  and reset flag
                              0236
    52   00000000'9F      DE  0236          MOVAL    @#PIO$GW_IIOIMPA,R2     ; Assume that this is an image file
                   22     E1  023D          BBC      #IFB$V_PPF_IMAGE,-      ; Now check the attached IFAB to see if
                07 69         023F                   (R9),10$                ;  we really have an image file
    52   00000000'9F      DE  0241          MOVAL    @#PIO$GW_PIOIMPA,R2     ; No, we have a process permanent file
                   04     E1  0248  10$:    BBC      #IMP$V_IORUNDOWN,-      ; If RMS rundown is in progress then
                09 62         024A                   (R2),20$                ;  set the I/O rundown event flag to
                              024C          $SETEF_S #IMP$C_IOREFN           ;  allow rundown to complete
                   04         0255  20$:    RET                              ; Dismiss AST
                              0256
       59   00FC C7       D0  0256  30$:    MOVL     NWA$L_THREAD(R7),R9     ; Pick up the IFAB/IRAB address that
                              025B                                       ;  was used to stall on
     00000000'EF   16         025B          JSB      RM$SETEFN               ; Retrieve event flag for the transfer
                52 8ED0       0261          POPL     R2                      ;  operation that we are stalled on
                              0264          $SETEF_S R2                      ;  and set it
     00000000'EF   17         026D          JMP      RM$THREADGO             ; Cause return from RM$STALL
```

```
                       0273          .SBTTL  PROCESS DAP MESSAGE PARSE FAILURE
                       0273
                       0273  ;++
                       0273  ; CASE 1 -- NT$DECODE_MSG failed to parse the received DAP message.
                       0273  ;
                       0273  ; Possible MACCODE values returned by NT$DECODE_MSG are DAP$_UNSUPPORT,
                       0273  ; DAP$_FORMAT, DAP$_INVALID, and DAP$_MSG_SYNC. The first is mapped to
                       0273  ; RMS$_SUP and the latter three are mapped into RMS$_BUG_DAP. Therefore:
                       0273  ;    (1) STS of FAB/RAB = RMS$_SUP or RMS$_BUG_DAP
                       0273  ;    (2) STV of FAB/RAB = DAP status code (prefixed by RMS facility code)
                       0273  ;
                       0273  ; Note: The STS value is returned in R0 less the RMS facility code.
                       0273  ;--
                       0273
                       0273  PARSE_FAILURE:                          ; Code segment of NT$RECEIVE
          1B A7   91   0273          CMPB    DAP$B_DCODE_MAC(R7),-   ; Branch if error is not
             0A        0276                  #DAP$_MSG_SYNC          ;  'message-out-of-sequence'
             06   12   0277          BNEQ    10$                     ;
   51     1A A7   9A   0279          MOVZBL  DAP$B_DCODE_MSG(R7),R1  ; Get message type number
             0A   11   027D          BRB     20$                     ; Join common code
   51     19 A7   9A   027F  10$:    MOVZBL  DAP$B_DCODE_FID(R7),R1  ; Get ID of field in error
          1A A7   F0   0283          INSV    DAP$B_DCODE_MSG(R7),-   ; Insert message type number
   51  06    06        0286                  #6,#6,R1                ;
          1B A7   F0   0289  20$:    INSV    DAP$B_DCODE_MAC(R7),-   ; Insert MACCODE error code
   51  04    0C        028C                  #12,#4,R1               ;
   0C A8     51   B0   028F          MOVW    R1,FAB$L_STV(R8)        ; Update STV field of FAB/RAB
             01   B0   0293          MOVW    #RMS__FACILITY,-        ; Add RMS facility code to value
          0E A8        0295                  FAB$L_STV+2(R8)         ;  in STV field of FAB/RAB
          1B A7   91   0297          CMPB    DAP$B_DCODE_MAC(R7),-   ; Did parse fail because a field/option
             02        029A                  #DAP$_UNSUPPORT         ;  received is not supported by RMS?
             06   13   029B          BEQL    30$                     ; Branch if yes
                       029D          RMSERR  BUG_DAP                 ; Declare Data Access Protocol error
                  05   02A2          RSB                             ; Exit with RMS code in R0
                       02A3  30$:    RMSERR  SUP                     ; Declare message unsupported
                  05   02A8          RSB                             ; Exit with RMS code in R0
                       02A9
                       02A9  ;++
                       02A9  ; This routine is called to report a Data Access Protocol error detected by
                       02A9  ; NT$SEARCH in processing the NAMETYPE field that was not detected by
                       02A9  ; NT$DECODE_MSG when the Name message was parsed.
                       02A9  ;--
                       02A9
                       02A9  NT$BUG_NAMETYPE::                       ; Entry point
             09   90   02A9          MOVB    #DAP$_INVALID,-         ; Field value is invalid
          1B A7        02AB                  DAP$B_DCODE_MAC(R7)     ;
             10   90   02AD          MOVB    #DAP$_NAMETYPE,-        ; Identify field
          19 A7        02AF                  DAP$B_DCODE_FID(R7)     ;
             C0   11   02B1          BRB     PARSE_FAILURE           ; Join common code
```

```
                          02B3          .SBTTL  PROCESS DAP STATUS MESSAGE FROM FAL
                          02B3
                          02B3   ;++
                          02B3   ; CASE 2 -- RMS received a DAP Status message from partner.
                          02B3   ;
                          02B3   ; The DAP status code may indicate:
                          02B3   ;    (1) success or failure of file operation attempted by the remote file system
                          02B3   ;    (2) the requested file operation is not supported by the remote system
                          02B3   ;    (3) a Data Access Protocol error detected at the remote system
                          02B3   ; In all cases, RMS will translate the DAP status code into an RMS completion
                          02B3   ; code that may also have an associated secondary status code.
                          02B3   ;
                          02B3   ; If the DAP status code maps directly into an RMS completion code, then:
                          02B3   ;    (1) STS of FAB/RAB = corresponding RMS status code
                          02B3   ;    (2) STV of FAB/RAB = 0 or supplementary information if any received from FAL
                          02B3   ; Or
                          02B3   ;    (1) STS of FAB/RAB = corresponding RMS status code
                          02B3   ;    (2) STV of FAB/RAB = secondary status code received if partner is VMS or
                          02B3   ;                         FAL$_DAPFAIL if partner is non-VMS
                          02B3   ;
                          02B3   ; If not, then one of the following indirect mappings will be made:
                          02B3   ;    (1) STS of FAB/RAB = RMS$_SUPPORT or RMS$_NETFAIL
                          02B3   ;    (2) STV of FAB/RAB = FAL status code mapped from DAP status code
                          02B3   ; Or
                          02B3   ;    (1) STS of FAB/RAB = RMS$_SUP, RMS$_NET, or RMS$_BUG_DAP
                          02B3   ;    (2) STV of FAB/RAB = DAP status code (prefixed by FAL facility code)
                          02B3   ;
                          02B3   ; Note: The STS value is returned in R0 less the RMS facility code.
                          02B3   ;--
                          02B3
                          02B3   STATUS_RETURN:                          ; Code segment of NT$RECEIVE
                          02B3
                          02B3          ASSUME  DAP$_PENDING EQ 0
                          02B3          ASSUME  DAP$_SUCCESS EQ 1
                          02B3          ASSUME  DAP$_UNSUPPORT EQ 2
                          02B3          ASSUME  DAP$_FILE_OPEN EQ 4
                          02B3          ASSUME  DAP$_FILE_XFER EQ 5
                          02B3          ASSUME  DAP$_WARNING EQ 6
                          02B3          ASSUME  DAP$_FILE_CLOS EQ 7
                          02B3          ASSUME  DAP$_FORMAT EQ 8
                          02B3          ASSUME  DAP$_INVALID EQ 9
                          02B3          ASSUME  DAP$_MSG_SYNC EQ 10
                          02B3
52   40 A7   0C   00   EF 02B3          EXTZV   #0,#12,DAP$W_STSCODE(R7),R2 ; Get MICCODE field
53   40 A7   04   0C   EF 02B9          EXTZV   #12,#4,DAP$W_STSCODE(R7),R3 ; Get MACCODE field
                          02BF          $CASEB  SELECTOR=R3-            ; Status returned by partner:
                          02BF                  DISPL=<-                ;
                          02BF                          FILE_ACCESS-    ; File operation pending
                          02BF                          FILE_ACCESS-    ; File operation successful
                          02BF                          UNSUPPORT-      ; Request not supported
                          02BF                          PROTOCOL-       ; Undefined value
                          02BF                          FILE_ACCESS-    ; Error related to opening a file
                          02BF                          FILE_ACCESS-    ; Error related to file transfer
                          02BF                          PROTOCOL-       ; Undefined value
                          02BF                          FILE_ACCESS-    ; Error related to closing a file
                          02BF   ;                      PROTOCOL-       ; Incorrect message format
                          02BF   ;                      PROTOCOL-       ; Invalid field value
```

```
                         02BF  ;                          PROTOCOL-               ;  Unexpected message for state
                         02BF                           >                         ;
                         02D3
                         02D3  ;+
                         02D3  ; Process Data Access Protocol error; RMS$_BUG_DAP will be returned.
                         02D3  ;-
                         02D3
                         02D3  PROTOCOL:                                          ; Dispatched here from CASE statement
                         02D3           RMSERR  BUG_DAP                           ; Declare Data Access Protocol error
              40 A7   B1 02D8           CMPW    DAP$W_STSCODE(R7),-               ; Check for invalid file name string
            90D2 8F      02DB                   #<<DAP$_INVALID@12>!-             ;  as this is not really a protocol
                         02DE                      <DAP$K_ACC_MSG@6>!-            ;  error, just a difference in file
                         02DE                      <DAP$_FILESPEC>!-              ;  specification formats between
                         02DE                   0>                                ;  systems
                 59   12 02DE           BNEQ    DAPCODE_TO_STV                    ; Branch if any other error
                         02E0           RMSERR  SYN                               ; Convert response to general
                 5E   11 02E5           BRB     DAPSTV_TO_STV                     ;  file name string syntax error
                         02E7                                                     ; Note that this prevents return of
                         02E7                                                     ;  RMS$_BUG_DAP in response to a
                         02E7                                                     ;  bad (invalid) file specification
                         02E7
                         02E7  ;+
                         02E7  ; Process operation not supported error condition; RMS$_SUPPORT (with a
                         02E7  ; secondary FAL status code from the NT$UNSUPPORTED conversion table) or
                         02E7  ; RMS$_SUP (with an associated DAP status code) will be returned.
                         02E7  ;-
                         02E7
                         02E7  UNSUPPORT:                                         ; Dispatched here from CASE statement
   52   00000800 8F   C0 02E7           ADDL2   #FAL$_OFFSET_B,R2                 ; Convert DAP MICCODE value to a FAL
                         02EE                                                     ;  message number (see RMSFALMSG.MSG)
   50      0000'CF    9E 02EE           MOVAB   W^NT$UNSUPPORTED,R0               ; Get address of conversion table
           51   80    3C 02F3  10$:     MOVZWL  (R0)+,R1                          ; Get next FAL message code
                11    13 02F6           BEQL    20$                               ; Branch if end of table
   53   51   0C   03  EF 02F8           EXTZV   #3,#12,R1,R3                      ; Obtain FAL message number (bits 3-14)
             53   52  B1 02FD           CMPW    R2,R3                             ; Does DAP status code correspond to
                         0300                                                     ;  this FAL message?
                 F1   12 0300           BNEQ    10$                               ; Branch if not--continue search
                         0302           RMSERR  SUPPORT                           ; Generate primary error code
                 25   11 0307           BRB     FALCODE_TO_STV                    ; Join common code
                         0309  20$:     RMSERR  SUP                               ; Generate catch-all primary error code
                 29   11 030E           BRB     DAPCODE_TO_STV                    ; Join common code
                         0310
                         0310  ;+
                         0310  ; Process status information (success or failure) of file operation attempted
                         0310  ; by the remote file system. The NT$DAP_TO_RMS conversion table will be used
                         0310  ; to map the DAP status code into a standard RMS completion code. RMS$_NETFAIL
                         0310  ; (with a secondary FAL status code) or RMS$_NET (with an associated DAP status
                         0310  ; code) will be returned if a direct mapping cannot be made.
                         0310  ;-
                         0310
                         0310  FILE_ACCESS:                                       ; Dispatched here from CASE statement
                         0310           RMSERR  NET                               ; Start with general file access error
      0000'8F      52 B1 0315           CMPW    R2,#NT$DAPRMSEND                  ; Branch if MICCODE value is too
                   1D 1A 031A           BGTRU   DAPCODE_TO_STV                    ;  large for conversion table
   50    0000'CF42    3C 031C           MOVZWL  W^NT$DAP_TO_RMS[R2],R0            ; Use MICCODE value as index into
                         0322                                                     ;  DAP-to-RMS conversion table
      1F 50   0D      E1 0322           BBC     #13,R0,DAPSTV_TO_STV              ; Branch if this is an RMS completion
```

```
                    0326                                                ;  code; not a FAL status code
      51    50  D0  0326                MOVL    R0,R1                   ; Copy FAL status code to R1
                    0329                RMSERR  NETFAIL                 ; Generate primary error code
                    032E                                                ; Fall thru to common code ...
                    032E
                    032E        ;+
                    032E        ; An RMS$_SUPPORT or RMS$_NETFAIL code has been generated and put in R0 with
                    032E        ; an associated FAL status code put in R1.
                    032E        ;-
                    032E
                    032E        FALCODE_TO_STV:                                 ;
   0C A8    51  B0  032E                MOVW    R1,FAB$L_STV(R8)        ; Put FAL status code from table in low
    01F7    8F  B0  0332                MOVW    #FAL$_FACILITY,-        ;  word of STV field of FAB/RAB
      0E    A8      0336                        FAB$L_STV+2(R8)         ;  and FAL facility code in high word
            05      0338                RSB                             ; Exit with RMS code in R0
                    0339
                    0339        ;+
                    0339        ; An RMS$_SUP, RMS$_NET, or RMS$_BUG_DAP code has been generated an put in R0.
                    0339        ; Each of these codes requires an associated DAP error code to be returned
                    0339        ; in the STV field of the FAB/RAB. This secondary code consists of the FAL
                    0339        ; facility code in the upper word and the DAP STSCODE value in the lower word.
                    0339        ;-
                    0339
                    0339        DAPCODE_TO_STV:                                 ;
      40    A7  B0  0339                MOVW    DAP$W_STSCODE(R7),-     ; Put DAP status code received in low
      0C    A8      033C                        FAB$L_STV(R8)           ;  word of STV field of FAB/RAB
    01F7    8F  B0  033E                MOVW    #FAL$_FACILITY,-        ;  and FAL facility code in high word
      0E    A8      0342                        FAB$L_STV+2(R8)         ;
            05      0344                RSB                             ; Exit with RMS code in R0
                    0345
                    0345        ;+
                    0345        ; The DAP Status message was mapped into an RMS completion code and put in R0.
                    0345        ; Update the STV field of the FAB/RAB as appropriate--some RMS codes require
                    0345        ; a secondary status code, some require a value, and others do not refer to it.
                    0345        ;-
                    0345
                    0345        DAPSTV_TO_STV:                                  ;
      4C    A7  D0  0345                MOVL    DAP$L_STV(R7),-         ; Put STV value received (if any)
      0C    A8      0348                        FAB$L_STV(R8)           ;  in STV field of FAB/RAB
   11 50    0E  E1  034A                BBC     #RMS$V_STVSTATUS,R0,20$ ; Branch if RMS code does not require
                    034E                                                ;  an associated status code in STV
                    034E                                                ;  (used by $GETMSG and $PUTMSG)
   05 67    34  E1  034E                BBC     #DAP$V_VAXVMS,(R7),10$  ; Use STV value from FAL only if
      0C    A8  D5  0352                TSTL    FAB$L_STV(R8)           ;  remote node is VMS and the value
            08  12  0355                BNEQ    20$                     ;  is non-zero
01F7A004    8F  D0  0357        10$:    MOVL    #FAL$_DAPFAIL,-         ; Otherwise, stuff a general FAL status
      0C    A8      035D                        FAB$L_STV(R8)           ;  code in STV
            05      035F        20$:    RSB                             ; Exit with RMS code in R0
```

```
                           0360          .SBTTL  NT$RMT_xxx ERROR REPORTING ROUTINES
                           0360
                           0360  ;++
                           0360  ; CASE 3 -- RMS generates an RMS$_SUPPORT error for a request not supported by
                           0360  ; the remote system based on the capabilities stated by FAL in its DAP
                           0360  ; Configuration message. The following status information is returned:
                           0360  ;    (1) STS of FAB/RAB = RMS$_SUPPORT
                           0360  ;    (2) STV of FAB/RAB = FAL status code
                           0360  ;
                           0360  ; Note: R8 contains the address of FAB/RAB on input.
                           0360  ;--
                           0360
                           0360  NT$RMT_ORG::                                        ; Unsupported value in ORG field
 51     C494 8F     3C     0360          MOVZWL  #<FAL$_ORG&^XFFFF>,R1      ; Generate secondary error code
             36     11     0365          BRB     RMT_COMMON                 ;
                           0367  NT$RMT_RFM::                                        ; Unsupported value in RFM field
 51     C49C 8F     3C     0367          MOVZWL  #<FAL$_RFM&^XFFFF>,R1      ; Generate secondary error code
             2F     11     036C          BRB     RMT_COMMON                 ;
                           036E  NT$RMT_RAT::                                        ; Unsupported value in RAT field
 51     C4A4 8F     3C     036E          MOVZWL  #<FAL$_RAT&^XFFFF>,R1      ; Generate secondary error code
             28     11     0373          BRB     RMT_COMMON                 ;
                           0375  NT$SUP_FOP::                                        ; Temporary
                           0375  NT$RMT_FOP1::                                       ; Unsupported value in FOP field
 51     C4EC 8F     3C     0375          MOVZWL  #<FAL$_FOP1&^XFFFF>,R1     ; Generate secondary error code
             21     11     037A          BRB     RMT_COMMON                 ;
                           037C  NT$RMT_FOP2::                                       ; Unsupported value in FOP field
 51     CE8C 8F     3C     037C          MOVZWL  #<FAL$_FOP2&^XFFFF>,R1     ; Generate secondary error code
             1A     11     0381          BRB     RMT_COMMON                 ;
                           0383  NT$RMT_RAC::                                        ; Unsupported value in RAC field
 51     C894 8F     3C     0383          MOVZWL  #<FAL$_RAC&^XFFFF>,R1      ; Generate secondary error code
             13     11     0388          BRB     RMT_COMMON                 ;
                           038A  NT$RMT_ROP::                                        ; Unsupported value in ROP field
 51     C8AC 8F     3C     038A          MOVZWL  #<FAL$_ROP&^XFFFF>,R1      ; Generate secondary error code
             0C     11     038F          BRB     RMT_COMMON                 ;
                           0391  NT$RMT_ACCFUNC::                                    ; Unsupported RMS service call
 51     C684 8F     3C     0391          MOVZWL  #<FAL$_ACCFUNC&^XFFFF>,R1;Generate secondary error code
             05     11     0396          BRB     RMT_COMMON                 ;
                           0398  NT$SUP_CTLFUNC::                                    ; Temporary
                           0398  NT$RMT_CTLFUNC::                                    ; Unsupported RMS service call
 51     C884 8F     3C     0398          MOVZWL  #<FAL$_CTLFUNC&^XFFFF>,R1;Generate secondary error code
                           039D  RMT_COMMON:                                         ; Common code
    0C A8    51     B0     039D          MOVW    R1,FAB$L_STV(R8)           ; Put associated FAL code in STV
      01F7 8F       B0     03A1          MOVW    #FAL$_FACILITY,-           ;  of FAB/RAB
        0E A8              03A5                  FAB$L_STV+2(R8)            ;
                           03A7          RMSERR  SUPPORT                    ; Declare primary error
                    05     03AC          RSB                                ; Exit with RMS code in R0
```

```
                      03AD  1034          .SBTTL  NT$LCL_xxx ERROR REPORTING ROUTINES
                      03AD  1035
                      03AD  1036 ;++
                      03AD  1037 ; CASE 4 -- RMS generates an RMS$_SUPPORT error for a request that it cannot
                      03AD  1038 ; support in a network context. The following status information is returned:
                      03AD  1039 ;    (1) STS of FAB/RAB = RMS$_SUPPORT
                      03AD  1040 ;    (2) STV of FAB/RAB = another RMS completion code
                      03AD  1041 ;
                      03AD  1042 ; Note: The secondary RMS error code cannot have an associated secondary code
                      03AD  1043 ;          of its own.
                      03AD  1044 ;
                      03AD  1045 ; Note: R8 contains the address of FAB/RAB on input.
                      03AD  1046 ;--
                      03AD  1047
                      03AD  1048 NT$LCL_RFM::                                    ; Unsuspported value in RFM field
                      03AD  1049          RMSERR  RFM,R1                          ; Generate secondary error code
           1A    11   03B2  1050          BRB     LCL_COMMON                      ;
                      03B4  1051 NT$LCL_FOP::                                    ; Unsupported value in FOP field
                      03B4  1052          RMSERR  FOP,R1                          ; Generate secondary error code
           13    11   03B9  1053          BRB     LCL_COMMON                      ;
                      03BB  1054 NT$LCL_ROP::                                    ; Unsupported value in ROP field
                      03BB  1055          RMSERR  ROP,R1                          ; Generate secondary error code
           0C    11   03C0  1056          BRB     LCL_COMMON                      ;
                      03C2  1057 NT$LCL_JOP::                                    ; Unsupported value in JOP field
                      03C2  1058          RMSERR  JOP,R1                          ; Generate secondary error code
           05    11   03C7  1059          BRB     LCL_COMMON                      ;
                      03C9  1060 NT$LCL_ENV::                                    ; Unsupported RMS service call
                      03C9  1061          RMSERR  ENV,R1                          ; Generate secondary error code
                      03CE  1062 LCL_COMMON:                                     ; Common code
     0C A8    51    B0   03CE  1063          MOVW    R1,FAB$L_STV(R8)                ; Put associated RMS code in STV
              01    B0   03D2  1064          MOVW    #RMS$_FACILITY,-                ;  of FAB/RAB
        0E A8            03D4  1065                  FAB$L_STV+2(R8)                 ;
                      03D6  1066          RMSERR  SUPPORT                         ; Declare primary error
                 05   03DB  1067          RSB                                     ; Exit with RMS code in R0
                      03DC  1068
                      03DC  1069 ;++
                      03DC  1070 ; Branch aids for exiting RMS.
                      03DC  1071 ;--
                      03DC  1072
                      03DC  1073 NT$EXRMS::                                      ; Exit RMS with failure code in R0
    00000000'EF  17   03DC  1074          JMP     RM$EXRMS                        ; Branch aid
                      03E2  1075 NT$EXSUC::                                      ; Exit RMS with success code in R0
    00000000'EF  17   03E2  1076          JMP     RM$EXSUC                        ; Branch aid
                      03E8  1077
                      03E8  1078
                      03E8  1079          .END                                    ; End of module
```

```
$$.PSECT_EP         = 00000000      DAP$_FILE_XFER      = 00000005
$$COUNT             = 000000C8      DAP$_FORMAT         = 00000008
$$RMSTEST           = 0000001A      DAP$_INVALID        = 00000009
$$RMS_PBUGCHK       = 00000010      DAP$_MSG_SYNC       = 0000000A
$$RMS_TBUGCHK       = 00000008      DAP$_NAMETYPE       = 00000010
$$RMS_UMODE         = 00000004      DAP$_PENDING        = 00000000
$$T1                = 00000000      DAP$_SUCCESS        = 00000001
DAP$B_BITCNT          00000035      DAP$_UNSUPPORT      = 00000002
DAP$B_DCODE_FID       00000019      DAP$_WARNING        = 00000006
DAP$B_DCODE_MAC       0000001B      DAPCODE_TO_STV        00000339 R     01
DAP$B_DCODE_MSG       0000001A      DAPSTV_TO_STV         00000345 R     01
DAP$B_DECVER          00000047      DECODE_DAP_MSG        000001F9 R     01
DAP$B_ECONUM          00000045      ERRSYS                00000116 R     01
DAP$B_FILESYS         00000043      ERRSYS1               0000017A R     01
DAP$B_FLAGS           00000031      ERRSYS2               00000222 R     01
DAP$B_LEN256          00000034      EXIT                  0000009E R     01
DAP$B_LENGTH          00000033      FAB$L_STV           = 0000000C
DAP$B_OSTYPE          00000042      FAL$_ACCFUNC        = 01F7C684
DAP$B_STREAMID        00000032      FAL$_CTLFUNC        = 01F7C884
DAP$B_TYPE            00000030      FAL$_DAPFAIL        = 01F7A004
DAP$B_USRNUM          00000046      FAL$_FACILITY       = 000001F7
DAP$B_USRVER          00000048      FAL$_FOP1           = 01F7C4EC
DAP$B_VERNUM          00000044      FAL$_FOP2           = 01F7CE8C
DAP$B_X_FIELD         00000024      FAL$_OFFSET_B       = 00000800
DAP$C_BLN             000000C0      FAL$_ORG            = 01F7C494
DAP$K_ACC_MSG       = 00000003      FAL$_RAC            = 01F7C894
DAP$K_BLN             000000C0      FAL$_RAT            = 01F7C4A4
DAP$K_STS_MSG       = 00000009      FAL$_RFM            = 01F7C49C
DAP$L_CMWA            00000030      FAL$_ROP            = 01F7C8AC
DAP$L_CRC_RSLT        00000020      FALCODE_TO_STV        0000032E R     01
DAP$L_DCODE_STS       00000018      FILE_ACCESS           00000310 R     01
DAP$L_MSG_MASK        0000001C      IFB$B_BID           = 00000008
DAP$L_RECNUM2         00000048      IFB$C_BID           = 0000000B
DAP$L_SSPWA           00000080      IFB$L_IOS           = 0000000C
DAP$L_STV             0000004C      IFB$L_NWA_PTR       = 0000003C
DAP$L_TEMP            00000090      IFB$V_PPF_IMAGE     = 00000022
DAP$M_BITCNT        = 00000008      IFB$W_CHNL          = 00000020
DAP$M_SEGMENT       = 00000040      IMP$C_ASYQIOEFN     = 0000001F
DAP$M_TMP1$         = 00000010      IMP$C_IOREFN        = 0000001E
DAP$M_TMP2$         = 00000080      IMP$V_IORUNDOWN     = 00000004
DAP$Q_DCODE_FLG       00000000      IO$M_INTERRUPT      = 00000040
DAP$Q_MSG_BUF1        00000008      IO$_READVBLK        = 00000031
DAP$Q_MSG_BUF2        00000010      IO$_WRITEVBLK       = 00000030
DAP$Q_STX             00000050      IRB$B_BID           = 00000008
DAP$Q_SYSCAP          00000028      IRB$C_BID           = 0000000A
DAP$Q_SYSPEC          00000038      IRB$L_IFAB_LNK      = 00000000
DAP$V_LENGTH        = 00000001      IRB$L_IOS           = 0000000C
DAP$V_MSGBLK        = 00000012      LCL_COMMON            000003CE R     01
DAP$V_VAXVMS        = 00000034      NT$BUG_NAMETYPE       000002A9 RG    01
DAP$W_BUFSIZ          00000040      NT$DAPRMSEND          ********    X  01
DAP$W_PARTNER         00000006      NT$DAP_TO_RMS         ********    X  01
DAP$W_RFA             00000042      NT$DECODE_MSG         ********    X  01
DAP$W_STSCODE         00000040      NT$EXRMS              000003DC RG    01
DAP$W_VERSION         00000004      NT$EXSUC              000003E2 RG    01
DAP$_FILESPEC       = 00000012      NT$INTERRUPT          0000013A RG    01
DAP$_FILE_CLOS      = 00000007      NT$LCL_ENV            000003C9 RG    01
DAP$_FILE_OPEN      = 00000004      NT$LCL_FOP            000003B4 RG    01
```

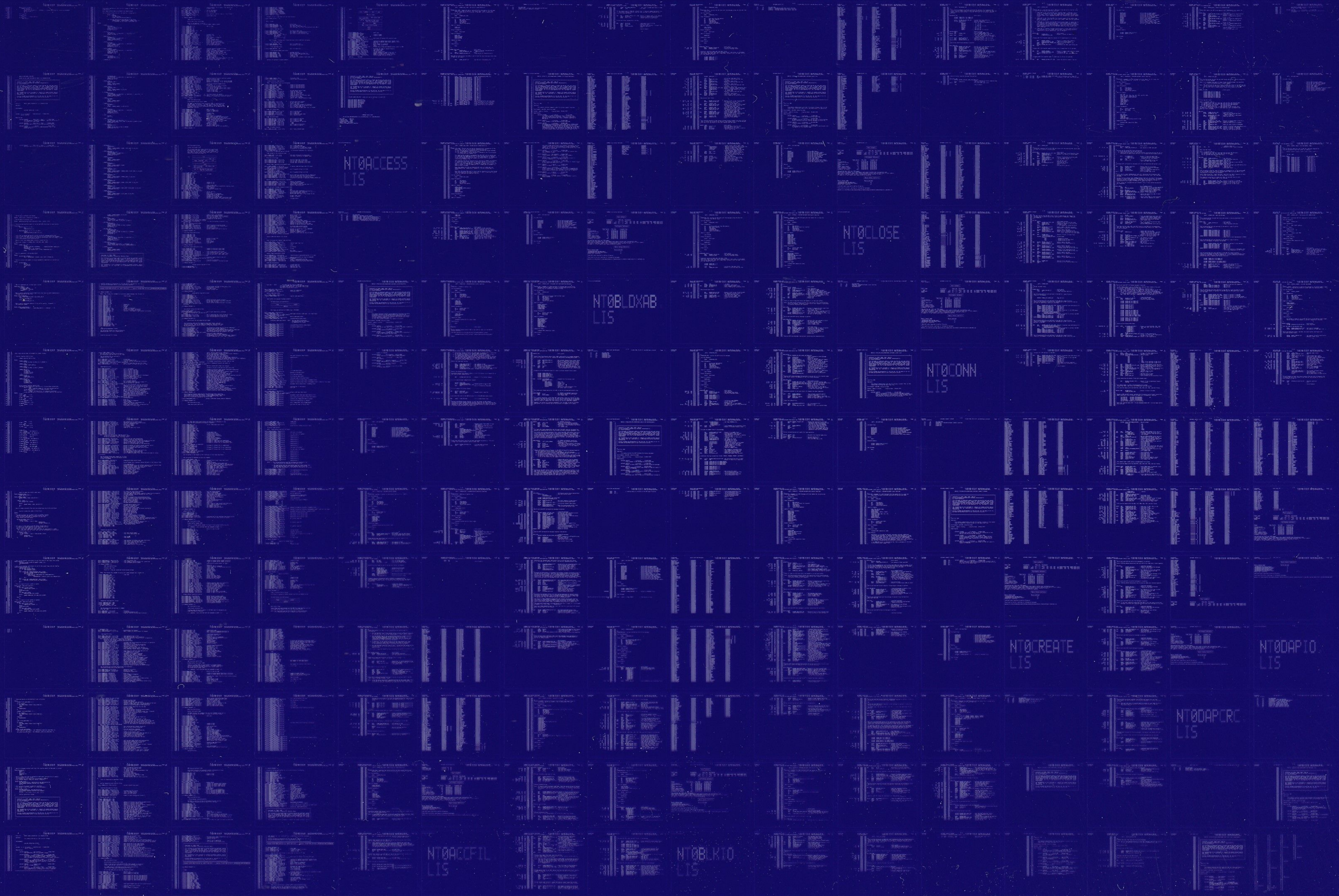
```
NT$LCL_JOP          000003C2 RG    01   NWA$Q_SAVE_DESC       00000120
NT$LCL_RFM          000003AD RG    01   NWA$Q_XLTBUF1         0000024C
NT$LCL_ROP          000003BB RG    01   NWA$Q_XLTBUF2         00000254
NT$RECEIVE          0000017D RG    01   NWA$Q_XMT             000000E8
NT$RMT_ACCFUNC      00000391 RG    01   NWA$T_ACSBUF          0000026C
NT$RMT_CTLFUNC      00000398 RG    01   NWA$T_AUXBUF          000005E0
NT$RMT_FOP1         00000375 RG    01   NWA$T_DAP             00000000
NT$RMT_FOP2         0000037C RG    01   NWA$T_INODEBUF        000004AC
NT$RMT_ORG          00000360 RG    01   NWA$T_ITM_ATTR        00000200
NT$RMT_RAC          00000383 RG    01   NWA$T_ITM_END         00000224
NT$RMT_RAT          0000036E RG    01   NWA$T_ITM_LST         00000200
NT$RMT_RFM          00000367 RG    01   NWA$T_ITM_MAXINDX     00000218
NT$RMT_ROP          0000038A RG    01   NWA$T_ITM_STRING      0000020C
NT$STALLAST         00000225 RG    01   NWA$T_NCBBUF          0000052C
NT$SUP_CTLFUNC      00000398 RG    01   NWA$T_NODEBUF         00000169
NT$SUP_FOP          00000375 RG    01   NWA$T_RCVBUF          000001A0
NT$TRANSMIT         00000000 RG    01   NWA$T_SCAN            00000100
NT$TRANSMIT_PKT     000000A5 RG    01   NWA$T_TEMP            00000120
NT$UNSUPPORTED      ********   X   01   NWA$T_XLTBUF1         000002AC
NWA$B_ALLXABCNT     0000011C            NWA$T_XLTBUF2         000003AC
NWA$B_DAP_RAC       000000C9            NWA$T_XMTBUF          000003C0
NWA$B_FILESYS       000000C5            NWA$V_FLUSH_BUF     = 00000001
NWA$B_KEYXABCNT     0000011D            NWA$V_LAST_MSG      = 00000000
NWA$B_NETSTRSIZ     0000016F            NWA$V_NODECODE      = 00000002
NWA$B_NODBUFSIZ     00000168            NWA$V_RCVAST        = 00000004
NWA$B_ORG           000000C6            NWA$V_RCVQIO        = 00000003
NWA$B_OSTYPE        000000C4            NWA$V_RCVSTALL      = 00000005
NWA$B_RFM           000000C7            NWA$W_BUILD           000000D2
NWA$B_RMS_RAC       000000C8            NWA$W_DAPBUFSIZ       000000CA
NWA$C_BLN           00000800            NWA$W_DIR_OFF         000000CC
NWA$K_BLN           00000800            NWA$W_DISPLAY         000000D0
NWA$L_ALLXABADR     00000100            NWA$W_FIL_OFF         000000CE
NWA$L_DATXABADR     00000104            NWA$W_JNLXABJOP       0000011E
NWA$L_DEV           000000C0            PARSE_FAILURE         00000273 R     01
NWA$L_FHCXABADR     00000108            PIO$A_TRACE           ********   X   01
NWA$L_KEYXABADR     0000010C            PIO$GW_IIOIMPA        ********   X   01
NWA$L_MSG_MASK      000000D4            PIO$GW_PIOIMPA        ********   X   01
NWA$L_PROXABADR     00000110            PROTOCOL              000002D3 R     01
NWA$L_RDTXABADR     00000114            RM$CHKAST             ********   X   01
NWA$L_SAVE_FLGS     00000128            RM$EXRMS              ********   X   01
NWA$L_SUMXABADR     00000118            RM$EXSUC              ********   X   01
NWA$L_THREAD        000000FC            RM$MAPERR             ********   X   01
NWA$L_XLTATTR       00000238            RM$SETEFN             ********   X   01
NWA$L_XLTBUFFLG     0000022C            RM$STALL              ********   X   01
NWA$L_XLTCNT        00000228            RM$STALLAST           ********   X   01
NWA$L_XLTMAXINDX    00000234            RM$THREADGO           ********   X   01
NWA$L_XLTSIZ        00000230            RMS$V_STVSTATUS     = 0000000E
NWA$Q_ACS           00000244            RMS$_BUG_DAP        = 00018444
NWA$Q_BIGBUF        00000170            RMS$_ENV            = 00018724
NWA$Q_BLD           000000F0            RMS$_FOP            = 0001853C
NWA$Q_FLG           00000000            RMS$_JOP            = 000187E4
NWA$Q_INODE         0000025C            RMS$_NET            = 0001874C
NWA$Q_IOSB          000000D8            RMS$_NETFAIL        = 0001C13C
NWA$Q_LNODE         00000160            RMS$_RFM            = 00018664
NWA$Q_LOGNAME       0000023C            RMS$_ROP            = 0001867C
NWA$Q_NCB           00000264            RMS$_SUP            = 000182D2
NWA$Q_RCV           000000E0            RMS$_SUPPORT        = 0001C144
```

```
RMS$_SYN                        = 000186D4
RMS$_SYS                        = 0001C10C
RMS$_FACILITY                   = 00000001
RMT_COMMON                        0000039D R       01
SETUP_QIO_PARAM                   00000122 R       01
STATUS_RETURN                     000002B3 R       01
SUC                               0000009B R       01
SYS$QIO                           ********   GX    01
SYS$SETEF                         ********   GX    01
TPT$L_NTDAP_DEC                   ********    X    01
TPT$L_NTDAP_ENC                   ********    X    01
TPT$L_NTDAP_RCV                   ********    X    01
TPT$L_NTDAP_XMT                   ********    X    01
UNSUPPORT                         000002E7 R       01

                                                 +----------------+
                                                 ! Psect synopsis !
                                                 +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (   0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
NK$NETWORK                      000003E8  (  1000.)  01 (   1.)    PIC   USR   CON   REL   GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE
$ABS$                           00000800  (  2048.)  02 (   2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE

                                              +------------------------+
                                              ! Performance indicators !
                                              +------------------------+

Phase                        Page faults    CPU Time         Elapsed Time
-----                        -----------    --------         ------------
Initialization                        32    00:00:00.05      00:00:01.28
Command processing                   134    00:00:00.64      00:00:03.58
Pass 1                               462    00:00:19.91      00:00:40.88
Symbol table sort                      0    00:00:02.64      00:00:02.90
Pass 2                               196    00:00:04.22      00:00:07.59
Symbol table output                   30    00:00:00.21      00:00:00.96
Psect synopsis output                  1    00:00:00.04      00:00:00.04
Cross-reference output                 0    00:00:00.00      00:00:00.00
Assembler run totals                 857    00:00:27.71      00:00:57.28

The working set limit was 1950 pages.
105813 bytes (207 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1839 non-local and 31 local symbols.
1079 source lines were read in Pass 1, producing 19 object records in Pass 2.
37 pages of virtual memory were used to define 36 macros.
```

```
                              +-----------------------------+
                              ! Macro library statistics !
                              +-----------------------------+

Macro library name                         Macros defined
------------------                         --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                   1
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                  21
_$255$DUA28:[SYSLIB]STARLET.MLB;2               10
TOTALS (all libraries)                          32

2087 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:NTODAPIO/OBJ=OBJ$:NTODAPIO MSRC$:NTODAPIO/UPDATE=(ENH$:NTODAPIO)+LIB$:RMS/LIB+EXECML$/LIB
```

0315 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0316 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY